قدم اول)

Dockerfile:

```
Dockerfile > ...

FROM alpine
RUN apk update
RUN apk add curl

Dockerfile > ...

RUN apk add curl

Dockerfile > ...

PROM alpine

Dockerfile > ...
```

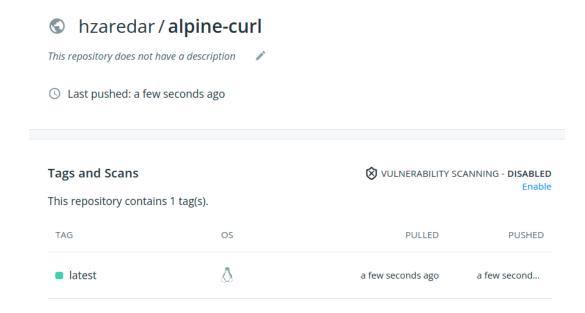
Building:

```
hossein@Hossein-Laptop:~/Cloud_HW2$ sudo docker build -t alpine-curl .
Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM alpine
   ---> c059bfaa849c
Step 2/3 : RUN apk update
   ---> Using cache
   ---> 51768a146bf8
Step 3/3 : RUN apk add curl
   ---> Using cache
   ---> 9c58b5f46980
Successfully built 9c58b5f46980
Successfully tagged alpine-curl:latest
```

Pushing:

```
hossein@Hossein-Laptop:~/Cloud_HW2$ sudo docker tag alpine-curl:latest hzaredar/alpine-curl:latest
hossein@Hossein-Laptop:~/Cloud_HW2$ sudo docker push hzaredar/alpine-curl:latest
The push refers to repository [docker.io/hzaredar/alpine-curl]
b8388bfcbadc: Pushed
860c4ba75eec: Pushed
8d3ac3489996: Mounted from hzaredar/cloud
latest: digest: sha256:14e2c683fdbd95db49c2f0102250a06067b3b09195e9646b24d7be88d7176f46 size: 950
```

Docker Hub:



Pulling:

```
hossein@Hossein-Laptop:~/Cloud_HW2$ sudo docker pull hzaredar/alpine-curl
Using default tag: latest
latest: Pulling from hzaredar/alpine-curl
Digest: sha256:14e2c683fdbd95db49c2f0102250a06067b3b09195e9646b24d7be88d7176f46
Status: Downloaded newer image for hzaredar/alpine-curl:latest
docker.io/hzaredar/alpine-curl:latest
```

Running:

```
<del>Hossein-Laptop:~/Cloud HW2</del>$ sudo docker run hzaredar/alpine-curl curl google.com
            % Received % Xferd Average Speed
                                                Time
                                                        Time
                                                                 Time Current
                                Dload Upload
                                                Total
                                                                 Left Speed
                                                        Spent
     219 100 <HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
                           0 0:00:03 0:00:02 0:00:01 73
```

قدم دوم)

پیادهسازی سرویس با Node.js انجام شده است.

فایل index.js مربوط به بخش اصلی برنامه است.

در فایل config.js، کانفیگها نوشته شده اند (و از index.js خوانده می شوند):

```
S2 > JS config.js > ...
1    var config = {}
2
3    config.time_api = "http://worldtimeapi.org/api/timezone/Asia/Tehran"
4    config.port = "8080"
5
6    export default config
```

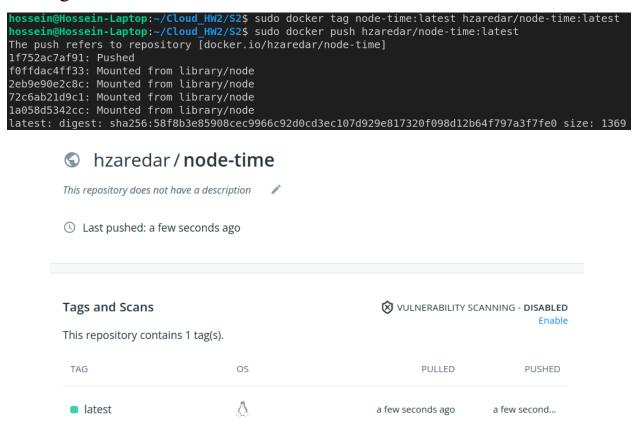
Dockerfile:

```
1 FROM node:alpine
2 COPY . /app
3 WORKDIR /app
4 CMD node index.js
```

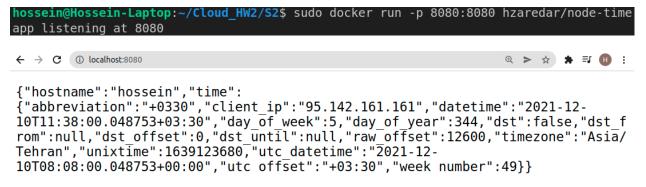
Building:

```
hossein@Hossein-Laptop:~/Cloud HW2/S2$ sudo docker build -t node-time .
Sending build context to Docker daemon 9.881MB
Step 1/4 : FROM node:alpine
---> bb1fcdaff936
Step 2/4 : COPY . /app
---> 2922eb0b92a8
Step 3/4 : WORKDIR /app
---> Running in 9d89511d9878
Removing intermediate container 9d89511d9878
---> e4385ae3bdd4
Step 4/4 : CMD node index.js
---> Running in 7481f3edfaa1
Removing intermediate container 7481f3edfaa1
---> ae540557dda7
Successfully built ae540557dda7
Successfully tagged node-time:latest
```

Pushing:



Running:



قدم سوم)

کانفیگها به صورت ConfigMap پیادهسازی شده است:

حال می توانیم در کامپوننت مربوط به deployment، کانفیگها را از طریق deployment و variable ها در اختیار کد سرور قرار دهیم. فایلهای service.yaml و service.yaml

سپس در config.js خواهیم داشت:

```
S3 > JS config.js > ...

1   var config = {}

2

3   config.time_api = process.env.TIME_API
4   config.port = process.env.PORT

5

6   export default config
```

اعمال فايلهاي yaml:

```
hossein@Hossein-Laptop:~/Cloud_HW2/S3$ kubectl apply -f config.yaml
configmap/time-config created
hossein@Hossein-Laptop:~/Cloud_HW2/S3$ kubectl apply -f deployment.yaml
deployment.apps/node-time-deployment created
hossein@Hossein-Laptop:~/Cloud_HW2/S3$ kubectl apply -f service.yaml
service/time-service created
```

Deployment ایجاد شده:

```
hossein@Hossein-Laptop:~/Cloud_HW2/S3$ kubectl get deployments

NAME READY UP-TO-DATE AVAILABLE AGE

node-time-deployment 2/2 2 2 2m36s
```

Pod های ایجاد شده:

hossein@Hossein-Laptop:~/Cloud_HW2/S3\$	kubectl	get pods		
NAME	READY	STATUS	RESTARTS	AGE
node-time-deployment-5c55c5d595-px4jl	1/1	Running	0	3m22s
node-time-deployment-5c55c5d595-zjwwf	1/1	Running	0	3m22s

سرویس ایجاد شده:

<pre>hossein@Hossein-Laptop:~/Cloud_HW2/S3\$ kubectl get services</pre>									
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE				
kubernetes	ClusterIP	10.96.0.1	<none></none>	443/TCP	3h4m				
time-service	NodePort	10.101.210.94	<none></none>	9000:30904/TCP	3m49s				

ارتباط ميان Pod ها و سرويس:

```
hossein@Hossein-Laptop:~/Cloud HW2/S3$ kubectl describe service time-service
Name:
                           time-service
Namespace:
                           default
Labels:
                           <none>
Annotations:
                           <none>
Selector:
                           app=node-time
Type:
                           NodePort
IP Family Policy:
                           SingleStack
IP Families:
                           IPv4
IP:
                           10.101.210.94
IPs:
                           10.101.210.94
Port:
                           <unset> 9000/TCP
TargetPort:
                           9000/TCP
NodePort:
                           <unset> 30904/TCP
Endpoints:
                           172.17.0.5:9000,172.17.0.6:9000
Session Affinity:
                           None
External Traffic Policy:
                           Cluster
Events:
                           <none>
```

همانطور که میبینیم، پادها در آدرسهای 172.17.0.5:9000 و 172.17.0.6:9000 شبکه داخلی کلاستر قرار دارند. سرویس به این دو پاد متصل است و خودش در آدرس 10.101.210.94 قرار گرفته است و از طریق پورت 9000 سرویس را ارائه میکند.

حال برای اینکه بتوانیم از بیرون به سرویس وصل شویم، میتوانیم با port-forwarding این کار را انجام دهیم:

```
hossein@Hossein-Laptop:~/Cloud_HW2/S3$ kubectl port-forward service/time-service 9000 Forwarding from 127.0.0.1:9000 -> 9000 Forwarding from [::1]:9000 -> 9000
```

```
{"hostname":"hossein","time":
{"abbreviation":"+0330","client_ip":"81.163.3.149","datetime":"2021-12-
10T21:02:14.059584+03:30","day_of_week":5,"day_of_year":344,"dst":false,"dst_f
rom":null,"dst_offset":0,"dst_until":null,"raw_offset":12600,"timezone":"Asia/
Tehran","unixtime":1639157534,"utc_datetime":"2021-12-
10T17:32:14.059584+00:00","utc_offset":"+03:30","week_number":49}}
```

⊕ > ☆ **★** ⊞ :

← → C ① localhost:9000

قدم چهارم)

در ابتدا نیاز است که به image قدم اول bash اضافه کنیم تا بلافاصله پس از اجرا شدن خاتمه نیابد و بتوانیم صورت interactive با آن کار کنیم:

```
1 FROM <u>alpine</u>
2 RUN apk update
3 RUN apk add curl
4 RUN apk add --no-cache bash
5 CMD bash
```

بعد از build کردن، می توانیم آن را در کلاستر k8s اجرا کنیم:

hossein@Hossein-Laptop:~/Cloud_HW2/S4\$ kubectl run -it acurl --image=hzaredar/alpine-curl-i
If you don't see a command prompt, try pressing enter.
bash-5.1# ■

اگر pod ها را لیست کنیم:

hossein@Hossein-Laptop:~/Cloud HW2/S3\$	kubectl	get pods		
NAME	READY	STATUS	RESTARTS	AGE
acurl	1/1	Running	0	34s
node-time-deployment-5c55c5d595-p96p9	1/1	Running	0	58s
node-time-deployment-5c55c5d595-t6zkn	1/1	Running	0	58s

مشاهده میکنیم که پاد جدید در کنار پادهای سرویس، در کلاستر اجرا شده است.

حال مى توانيم از كامند curl استفاده كنيم تا به سرويس متصل شويم:

bash-5.1# curl 10.101.190.220:9000 {"hostname":"hossein", "time":{"abbreviation":"+0330", "client_ip":"5.211.197.40", "datetime":"2021-12-10 T22:29:12.485185+03:30", "day_of_week":5, "day_of_year":344, "dst":false, "dst_from":null, "dst_offset":0, "dst_until":null, "raw_offset":12600, "timezone":"Asia/Tehran", "unixtime":1639162752, "utc_datetime":"2021 -12-10T18:59:12.485185+00:00", "utc_offset":"+03:30", "week number":49}}bash-5.1#